Table of Contents

1 Welcome to BTracer2....1
1.1 BTracer2 vs BTracer....1
1.2 BTracer2 vs classic Voxelstack/Fiji....2
2 Licenses....2
2.1 BTracer2x64 NON-COMMERCIAL-License....2
2.2 BTracer2x64 COMMERCIAL-License....3
3 BTracer2-Options....3
3.1 Integrated BTracer2....3
3.1.1 Limitations....3
3.1.2 Options....3
3.1.3 *.btrace2-files....5
4 External 64Bit-companion-application BTracer2x64....5
4.1 Why not Mandelbulbx64?....6
4.2 Automatic mesh reduction (experimental feature)....6
4.3 Mesh reduction sequences....6
4.4 Options....7
5 Known limitations/work in progress....8
6 Contact the author....8
6.1 How to not contact the author....9

1 Welcome to BTracer2

BTracer2 (“Bulb Tracer 2”) is the 2nd generation of an integrated mesh generator for Mandelbulb3D.

It was completely rewritten to achieve the following goals:

- much higher mesh quality (with less stepping artifacts)
- generate huge meshes quickly
- support for colored meshes (uv-coordinates or vertex colors)
- create high-resolution meshes with never-seen detail (which makes it necessary to circumvent the 32bit-limit of the Mandelbulb3D application)
- make it simple to use

1.1 BTracer2 vs BTracer

BTracer2 has the following advantages over BTracer:

- highly improved mesh quality
- optional creation of color information (uv-coordinates or vertex colors)
- (much)improved performance

- streamlined UI, improved UI performance
- optional “live”-OpenGL preview (to preview the mesh before actually generating it, may be not a good option for old/slow machines, so it is optional)
- support for meshes of (practical) unlimited size, enabled by an external 64bit-companion application (BTracer2x64) - for example my machine has 32 GB of memory and allows me to create meshes with a polygon count of 200 millions
- support for tracing-ranges and creation of closed meshes
- experimental feature: automatic mesh reduction (requires the external 64bit-companion application): reduces polygon count at parts of the mesh with lower detail level (e.g. smooth surfaces), and leaves true details

1.2 BTracer2 vs classic Voxelstack/Fiji

The still integrated Voxelstack/Fiji-solution is very solid, but also very limited in comparison to BTracer2:

- the reachable high “resolutions” in Voxelstacks are much overrated: each voxel can only be “filled” or “empty” (white pixel or black pixel in a slice), while voxels in BTracer2 can have an unlimited number of states. This leads to a much higher level of detail in BTracer2 at the same volumetric resolution, in comparison to Voxelstacks.
- Voxelstacks are vulnerable for stepping artifacts in smooth surfaces (like a sphere), BTracer2 isn’t
- much improved generation speed, multi-threading, integrated preview
- support for coloring

2 Licenses

If you are just using Mandelbulb3D with the integrated BTracer2 (which can really create very awesome meshes in high quality, very quickly), you can completely ignore the license topic. Mandelbulb3d itself never had any restrictions, and this is still the case, and will always be the case.

2.1 BTracer2x64 NON-COMMERCIAL-License

If you are using the external 64bit-addon (“BTracer2x64”) for non-commercial projects, you can just use the software as you downloaded it. There is not any restriction.

But, if you like the software, you are welcome to support the development by making a donation at my official blog (http://www.andreas-maschke.com/?page_id=1401).

2.2 BTracer2x64 COMMERCIAL-License

Please contact me at thargor6@googlemail.com if you want to use the external 64bit-addon ("BTracer2x64") for commercial projects. You are not allowed to use the NON-COMMERCIAL version in such projects and must obtain a separate license. If you have already have supported the development, you may get discounts etc. - just don't hesitate to contact me.

3 BTracer2-Options

3.1 Integrated BTracer2

3.1.1 Limitations

The integrated BTracer2 has almost any options of the "full solution", except:

- unlimited mesh size (because it is limited by the available memory, a 32bit application can handle) – but you can still create meshes in a very high quality, and this very quickly
- automatic mesh reduction

3.1.2 Options

- **Offset (X, Y, Z)**: the offset of the fractal inside the tracing-range
- **Scale**: the size of the fractal inside the tracing-range
  - **Suggestion**:
    - when you import a fractal and you "see nothing", decrease the Scale-parameter quickly, until you seem some "starting voxels" in the preview
    - from this point you can re-position and re-scale
    - when you still "see nothing", decrease **Preview Size** and increase **DE Stop** parameter (both parameters a described below)
- **Angle(X, Y, Z)**: the orientation of the fractal inside the tracing-range. Please note, that the angles are given in degrees.

- **Volumetric resolution**: number of voxels in each direction (X, Y, Z). Please note, that
  - you will need a much lower volumetric resolution than using voxel-stacks (voxels-stacks only use two colors, black and white, to indicate the state of a voxel. BTracer2-voxels can have much more states)
  - doubling this parameter leads to 8 times (2*2*2) computation time and memory consumption. E.g., going from 128 to 1280 leads to a factor of 1000 (10 * 10 * 10)!

- **Surface Detail**: this is the most important parameter for creating nice meshes (with nice detail and without stepping artifacts). Unfortunately, you must find the optimum value for each fractal by yourself.
  - **Rule of thumb**:
    - when you are missing details, the value is too small
    - when you see stepping artifacts, the value is too high
    - values in the range of 0.01 … 2.0 are common
  - **Suggested workflow**:
    - before using the 64Bit-companion-app, find the best value by using the integrated BTracer2, because it is much faster.
    - a volumetric resolution of 256x256x256 up to 512x512x512 should be sufficient to judge the mesh quality.

- **Calculate Colors**: Indicate if BTracer2 should calculate color-information. While it is not a big deal for meshes with low volumetric resolution, it is recommended to turn it off, when calculating large meshes (volumetric resolutions of 1000x1000x1000 or higher) and not caring about color-information at all. It takes more time and consumes memory, which may be a lot of memory on huge meshes.

- **Close Mesh**: Indicate if BTracer2 should close the mesh along the tracing-boarders

- **Trace Range (X, Y, Z)**: The trace range in percent. This is useful if you want to trace only a “slide” or “pillar” of the whole fractal.

- **Save Type**: Indicates the type of output of BTracer2:
  - **Mesh with vertex-colors (*.ply)**: the default mesh-output. Can contain color-information as vertex-colors (actual rgb-values). Unfortunately, this format is not well supported by all 3d-applications.
  - **Mesh with uv-colors (*.obj)**: an alternative mesh-output. Can contain color-information as uv-coordinates (requires a separate texture to actually render colors). May be a good alternative, when your 3d-application does not support the *.ply-files and/or vertex-colors (uv-coordinates are nearly supported by any 3d-application).
  - **BTracer2 Cache (*.btr2cache)**: Tracing-data to be used by the 64Bit-companion-application to actually generate meshes.

        - In comparison to generating meshes (which must fit completely in memory), Mandelbulb3D can generate *.btr2cache-files for meshes of any size.
        - But, beware, that doubling the volumetric resolution leads to 8 times (2*2*2) computation time and memory consumption. Again: going from 128 to 1280 leads to a factor of 1000 (10 * 10 * 10) - so *.btr2cache-files can become very large
        - Actually, *.btr2cache-”files” are directories, which contain many files
    - **Don’t save, only preview**: useful, when you have a slow hard drive, where saving time does matter, and you just want to create preview-meshes to play with the **Surface Detail** parameter

- **Preview Size**: number of preview voxels. More (and smaller) voxels show much more detail, but need more computation time. On some fractals, the preview can become invisible, when increasing the **Preview Size**. In this case you need to increase the **DE stop** size to make it visible again.
- **DE Stop**: Threshold value for calculation the quick preview.
    - when the preview becomes invisible, you should increase this value
    - when the shape has to less detail or looks “too round”, sou should decrease this value
- **Auto update preview**: indicate if the preview should automatically update when ychanging parameters
- **Use OpenGL (for preview)**: indicate if you want to use “live”-OpenGL-preview. It is slower than the classic voxel-based preview, but gives you much more accurate results. And you can move/scale/rotate the mesh in realtime as you know it from the OpenGL-mesh-display
- **OpenGL Preview**: indicate, if the final generated mesh should be displayed in an OpenGL window. Should only be turned off when your machine has problems to operate OpenGL.
- **Max. Vertices Preview**: maximum number of vertices, displayed in the **OpenGL Preview**. When the mesh has more vertices, a reduced mesh is created for preview. This is to use less memory (the OpenGL-preview needs to hold all the vertices etc. in memory of the 32bit application) and decrease computation time of the preview.

3.1.3 *.btrace2-files

All the options, together with the Fractal params, can be saved as *.btrace2-file and later be restored.

4 External 64Bit-companion-application BTracer2x64

The 64Bit-companion-application was written to circumvent the 32Bit-limit of the Mandelbulb3D application.

The basic idea is to use Mandelbulb3d (32bit) to do the fractal calculations, let Mandelbulb3D write the results to disc and then use a 64Bit-application to load this information and generate meshes.

Of course, it is not simple like that. The information on disc can become very large (terabytes), so it can not just be loaded into memory of the 64Bit-application. Also, there is a lot of parallel processing involved in order to decrease computation time.

4.1 Why not Mandelbulbx64?

As it was discussed in the community so many times, there is no practical way to create a 64Bit-Mandelbulb3D-application, without essentially completely rewriting it.

Even the creation of the much more smaller BTracer2x64-application took a lot of effort and time (we speak here not about hours, but about weeks and months).
So, creating a Mandelbulbx64-application (in the same quality as Mandelbulb3D) would probably take years, so it is very unlikely to happen.

4.2 Automatic mesh reduction (experimental feature)

Many fractals contain different levels of detail (e. g., a smooth sphere-shape, with a lot of little small spheres on top of it). A constant mesh resolution does not fit this well and leads to unnecessary vertices and faces, which lead to very large files.

So, the basic idea is to reduce the mesh in an adaptive way, i. e. to reduce mesh details, where they would not be seen.

4.3 Mesh reduction sequences

Because we speak about very huge meshes (a face count of 200 millions is typical), it is very tedious to “try out” the right settings for automatic mesh-reduction:

- is the mesh reduction-threshold too high, then we loose too many details
- is the mesh-reduction-threshold too low, the file remains too large (this is not a problem in theory, but a practical problem: many 3d-software, which I have tested, just crashes when trying to import files at a certain size)

So, the basic idea is not to create a simple reduced mesh, but to create a range of meshes and allow the user then to pick the “right” one.

So, you can specify a sequence of reduction levels, and BTracer2x64 will create a mesh for each of this reduction levels. This is much more efficient than creating a huge mesh and then reducing it multiple times, because:

- the initial creation step must be done only once (this can take hours)
- BTracer2x64 can use an already reduced mesh to generate a further reduces mesh

4.4 Options

- **Volumetric resolution**: number of voxels in each direction (X, Y, Z) in the cache.

- **Surface Detail**: this is the most important parameter for creating nice meshes (with nice detail and without stepping artifacts). Unfortunately, you must find the optimum value for each fractal by yourself.
  - **Rule of thumb**:
    - when you are missing details, the value is too small
    - when you see stepping artifacts, the value is too high
    - values in the range of 0.01 … 2.0 are common
  - **Suggested workflow**:
    - before using the 64Bit-companion-app, find the best value by using the integrated BTracer2, because it is much faster.
    - a volumetric resolution of 256x256x256 up to 512x512x512 should be sufficient to judge the mesh quality.
  - Please note, that you can change this parameter even after creating the cache. You, you can create try out different settings on top of the same cache.

- **Calculate Colors**: Indicate if BTracer2 should calculate color-information. While it is not a big deal for meshes with low volumetric resolution, it is recommended to turn it off, when calculating large meshes (volumetric resolutions of 1000x1000x1000 or higher) and not caring about color-information at all. It takes more time and consumes memory, which may be a lot of memory on huge meshes.
  - Of course, generation of color-information can be disabled, even if the cache does contain color-information
- **Close Mesh**: Indicate if BTracer2 should close the mesh along the tracing-boarders
  - Please note, that you can change this parameter even after creating the cache. You, you can create try out different settings on top of the same cache.

- **Trace Range (X, Y, Z)**: The trace range in percent. This is useful if you want to trace only a “slide” or “pillar” of the whole fractal.
  - Please note, that you can change this parameter even after creating the cache. You, you can create try out different settings on top of the same cache.

- **Save Type**: Indicates the type of output of BTracer2:
  - **Mesh with vertex-colors (*.ply)**: the default mesh-output. Can contain color-information as vertex-colors (actual rgb-values). Unfortunately, this format is not well supported by all 3d-applications.
  - **Mesh with uv-colors (*.obj)**: an alternative mesh-output. Can contain color-information as uv-coordinates (requires a separate texture to actually render colors). May be a good alternative, when your 3d-application does not support the *.ply-files and/or vertex-colors (uv-coordinates are nearly supported by any 3d-application).

- **Mesh reduction sequence**: Please see the separate section about Mesh reduction sequences for a brief explanation of the meaning of this parameter.
  - If you don't want to use this feature, just leave the field blank
  - Please always enter the mesh-reduction values in increasing order (in order to avoid strange results)
  - meaning of the numerical value:
    - a value of 0 means no reduction at all
    - the higher the value, the higher the reduction amount
    - a value of 5.0 usually leads to a very high reduction of about 5%
    - a value of 2.5 usually leads to a low reduction of about 50%

5 Known limitations/work in progress

Currently, the following features are known to have limitations:

- the "live"-OpenGL-preview could be faster
- the color-calculation of meshes does not always lead to nice results
- the algorithm for automatic meh reduction needs more refinement

6 Contact the author

Please contact me for feedback/suggestions etc. per email: thargor6@googlemail.com

6.1 How to not contact the author

Many of you know that I sometimes submit fractal artworks, e. g. at Facebook. Please do not use the comment function at my art to submit any bug reports etc. to the software. It strangely happens very often, but I hate it.

Please do not send me personal/primate messages at any social platform. I may read it by occasion, but usually I will not have the time to do so.